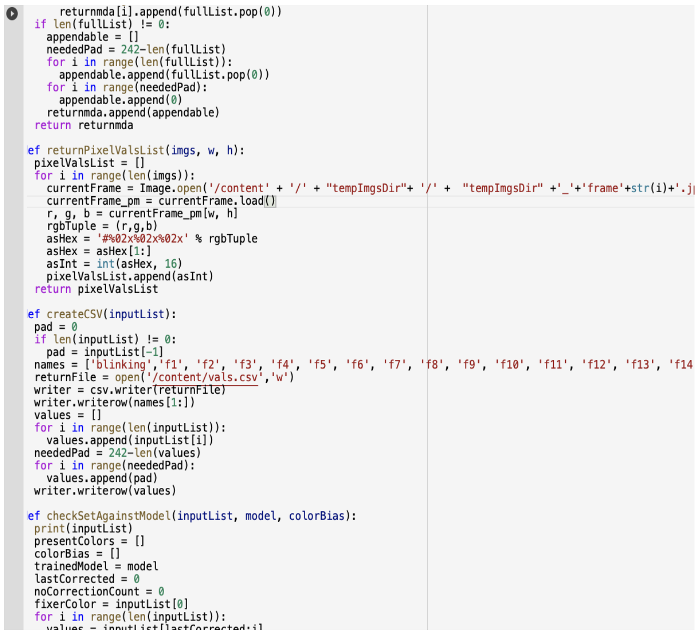
Detecting and Correcting Harmful Stimuli in Video Content to Provide Safer Viewing Experiences to Photosensitive and Seizure-Prone Individuals



```
      returnmda[i].append(fullList.pop(0))
  if len(fullList) != 0:
    appendable = []
    neededPad = 242-len(fullList)
    for i in range(len(fullList)):
      appendable.append(fullList.pop(0))
    for i in range(neededPad):
      appendable.append(0)
    returnmda.append(appendable)
  return returnmda

ef returnPixelValsList(imgs, w, h):
  pixelValsList = []
  for i in range(len(imgs)):
    currentFrame = Image.open('/content' + '/' + "tempImgsDir"+ '/' +  "tempImgsDir" +'_'+'frame'+str(i)+'.j
    currentFrame_pm = currentFrame.load()
    r, g, b = currentFrame_pm[w, h]
    rgbTuple = (r,g,b)
    asHex = '#%02x%02x%02x' % rgbTuple
    asHex = asHex[1:]
    asInt = int(asHex, 16)
    pixelValsList.append(asInt)
  return pixelValsList

ef createCSV(inputList):
  pad = 0
  if len(inputList) != 0:
    pad = inputList[-1]
  names = ['blinking','f1', 'f2', 'f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10', 'f11', 'f12', 'f13', 'f14
  returnFile = open('/content/vals.csv','w')
  writer = csv.writer(returnFile)
  writer.writerow(names[1:])
  values = []
  for i in range(len(inputList)):
    values.append(inputList[i])
  neededPad = 242-len(values)
  for i in range(neededPad):
    values.append(pad)
  writer.writerow(values)

ef checkSetAgainstModel(inputList, model, colorBias):
  print(inputList)
  presentColors = []
  colorBias = []
  trainedModel = model
  lastCorrected = 0
  noCorrectionCount = 0
  fixerColor = inputList[0]
  for i in range(len(inputList)):
    values = inputList[lastCorrected:i]
```

Ethan Effendi, 8th Grade
11/16/2022

Table of Contents

Abstract

In today's internet age, advanced technologies entertain and inform visually via mediums such as television and advanced virtual reality goggles. However, when intense stimuli appear in video data, viewers with conditions such as photosensitive epilepsy experience discomfort, headache, or even fatal seizures. Fortunately, entertainment products, such as Disney's *Incredibles 2* or Berzerk Studio's *Just Shapes and Beats*, are accompanied by seizure warnings and sometimes, safer variations. In environments where photosensitive individuals may unknowingly watch media not yet corrected for safety, however, a workflow which automates the creation of safer variations may help reduce epilepsy's annual 1000 person death toll and increase internet accessibility. The development of this workflow was approached by training a binary time series classification machine learning model. The training data used consisted of two hundred forty-two frames as features, with the label being whether or not each set represented blinking. When the workflow processes a video, the video is decomposed into pixel locations from which an array of color values may be derived. While a loop iterates through each array, the trained model is applied, makes decisions, and based on decisions, corrects particular intervals or preserves their values. After experimentation and various trials, a monochrome, four-by-four, minute-long video blinking at a frequency of 29.5 Hz inputted through the workflow was determined to produce an output where all blinking was corrected (0 Hz). Similarly, the same four-by-four, minute-long video possessed by green (#f4ae3d) and orange (#86b953) pixels yielded identical successful results. However, when the trial was repeated using the colors #d23a20 and #4924ac, an average of 14.75 Hz was the pixels' collective blinking frequency, possibly due to the trained model's unfamiliarity with such color values.

Background Research

The objective of this year's engineering project is to develop a software workflow which accepts video media as input and reduces any stimuli harmful towards photosensitive viewers. The question being tested is, "Is it possible to automate the creation of harmful videos' safer variations, particularly using machine learning methods?" A rational hypothesis is that it is possible to automate the correction of harmful stimuli in such videos using tabular (time series) binary classification models. The interesting aspect of this question lies in computers' capability to not only operate the spreadsheets we use daily, or exist as a source of entertainment, but also to work for the welfare of people. Machine learning is an exciting, relatively novel topic that has the capability to work impressive tricks: it can draw accurate illustrations depicting a given prompt or even unlock your phone using your face. However, above impressive tricks, it is integral that engineers continue trying to discover integral environments, such as health, education, or accessibility, where these theories may make positive differences in societies today. The testable question at hand experiments with machine learning's capabilities in relation to making the digital world a more accessible place for more people. Video media has the ability to visually transport viewers across the world, connect loved ones in different countries, and entertain people in cinema seats. To ensure more people, particularly those with photosensitive epilepsy, can enjoy these benefits of video data, this project aims to reduce harmful stimuli.

Throughout the years, the study of machine learning (ML) has evolved from its roots at International Business Computer (IBM). Credits for the invention of machine learning often go to a former IBM employee, Arthur Samuel, who worked on the 701 computer and spurred ML research by examining how computers may be trained to play the game of checkers. Samuel observed that the more his software played the game, it learned from experience, thus becoming a more competent opponent. This program was completed on the 701 machine in the

Poughkeepsie Laboratory, and its demonstration increased IBM's stock price value by fifteen percent. The concepts Samuel put forth during his work at IBM remain relevant today: the more 'experience' provided to a machine learning program (the data we allow it to learn from) improves its performance in its respective field. The idea of stimulating computers' intelligence by teaching it with data has grown since Samuel introduced his checkers program, with Frank Rosenblatt's development of artificial neural networks, and continued improvements at numerous institutions. Today, experimentation with machine learning is freely available to anyone with a computer and internet connection, and is made particularly accessible with the help of ML frameworks such as Tensorflow and Pytorch. Tensorflow, introduced in 2015 by Google, a company based in Mountain View, California, allows one to train their machine learning models, and make inferences after training. This project's machine learning aspects are made possible by this framework, particularly its Gradient Boosted Tree Model. This model makes use of decision trees, another machine learning concept brought forth by the research following that of Arthur Samuel, to improve intelligence by examining given data. Today, machine learning concepts are used to make useful predictions with various data types including tabular data (data typically placed in rows and columns), video data, image data, and audio data. This makes unlocking a cell phone using an image of your face possible, as well as detecting cancer in medical images. Other products may use machine learning to stitch together videos from several photos, make predictions out of tabular data representing animal populations, or even digitize and improve the quality of old family photos. Research related to the problem this project attempts to solve has also spanned throughout decades. Worldwide, doctors have researched conditions such as photosensitive epilepsy, where stimuli such as blinking may prove harmful. One report states that in 1997, an episode of a popular show, Pokémon, displayed a rocket launch scene. The rocket launch scene displayed a background blinking at 12.5 Hz between red and blue color values. Resultantly 685 children who watched the episode were taken to hospitals, some being diagnosed with seizures.

In pursuing an answer to this project's testable question, I hope to gain more insight on software and machine learning's abilities to make the internet and the use of computers more accessible for people with particular conditions. Furthermore, I aspire to increase my knowledge of machine learning, understand the topics which bring it life, and ways to access existing ML technologies. The work done on this project will challenge my problem solving skills and increase my research perseverance and stamina.

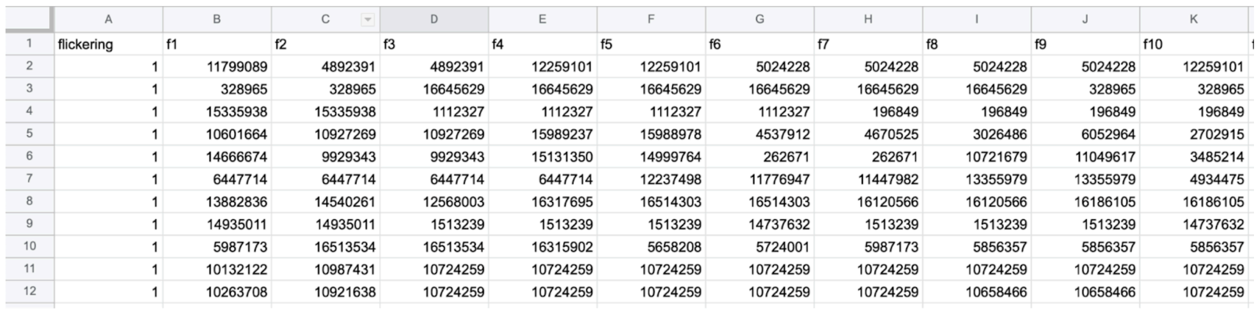
Materials and Procedures

Materials Used

1. One computer to run all programs.
2. One trained machine learning model taught to recognize flickering and flashing.
3. Ten videos made with particular instances of flashing and flickering. The frequency at which blinking occurs in each video should be known (measured in Hz).

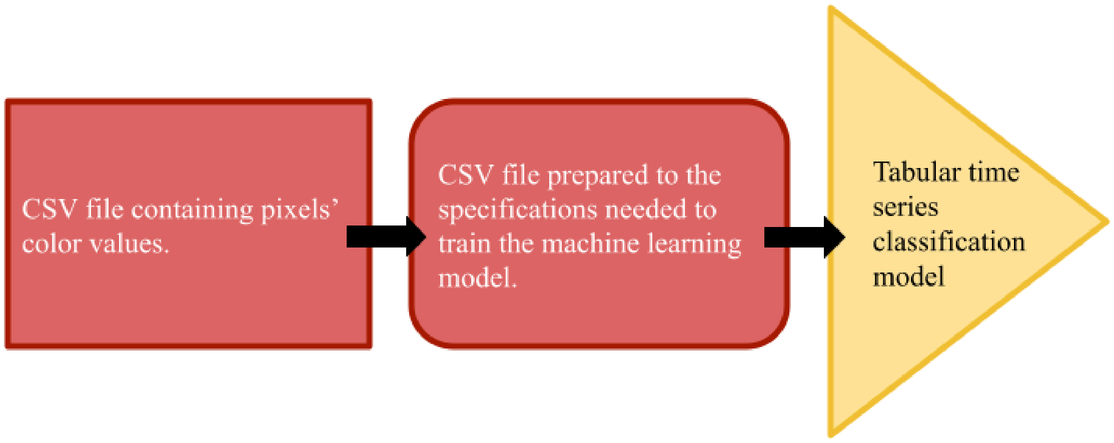
Procedure

1. Collect the data used to train the machine learning model as tabular data in a spreadsheet. Once all this data is collected, download it as a comma separated value file. The process of obtaining values to turn into tabular data should be automated: first, we obtain flickering pixels, obtain numerical values for the color of these pixels, and use Google Sheets APIs to automatically input these values into Google Sheets to be downloaded later on.



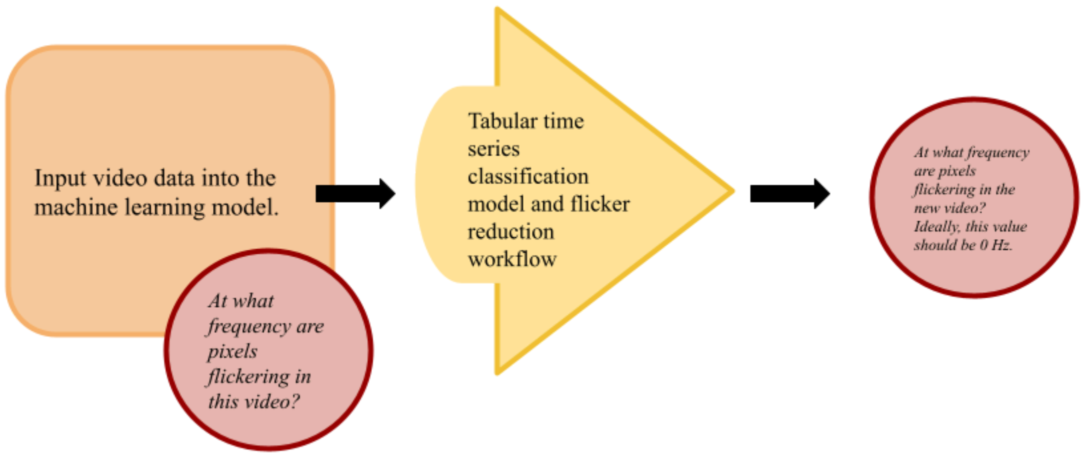
	A	B	C	D	E	F	G	H	I	J	K
1	flickering	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
2	1	11799089	4892391	4892391	12259101	12259101	5024228	5024228	5024228	5024228	12259101
3	1	328965	328965	16645629	16645629	16645629	16645629	16645629	16645629	328965	328965
4	1	15335938	15335938	1112327	1112327	1112327	1112327	196849	196849	196849	196849
5	1	10601664	10927269	10927269	15989237	15988978	4537912	4670525	3026486	6052964	2702915
6	1	14666674	9929343	9929343	15131350	14999764	262671	262671	10721679	11049617	3485214
7	1	6447714	6447714	6447714	6447714	12237498	11776947	11447982	13355979	13355979	4934475
8	1	13882836	14540261	12568003	16317695	16514303	16514303	16120566	16120566	16186105	16186105
9	1	14935011	14935011	1513239	1513239	1513239	14737632	1513239	1513239	1513239	14737632
10	1	5987173	16513534	16513534	16315902	5658208	5724001	5987173	5856357	5856357	5856357
11	1	10132122	10987431	10724259	10724259	10724259	10724259	10724259	10724259	10724259	10724259
12	1	10263708	10921638	10724259	10724259	10724259	10724259	10724259	10658466	10658466	10724259

2. Once all data needed is prepared as a comma separated value file, prepare it and train a machine learning model using it.


CSV file containing pixels' color values.
CSV file prepared to the specifications needed to train the machine learning model.
Tabular time series classification model

3. Couple the trained model with a workflow which uses the model to detect and thus correct blinking. The frequency at which the initial video of blinking, the frequency at which the output video is blinking (ideally, this value should be 0 Hz, meaning that all blinking is gone), and the time which it took for the workflow to process the input video data should all be recorded.
    a. The first test using this method should display the relation between the blinking frequency of monochrome videos to the blinking frequency of their corrected copies after being corrected by the workflow.
    b. The second test using this method should relate the blinking of different colors to the blinking frequencies of the corrected videos given that the original videos

blink at 29.5 Hz.


Input video data into the machine learning model.
At what frequency are pixels flickering in this video?
Tabular time series classification model and flicker reduction workflow
At what frequency are pixels flickering in the new video? Ideally, this value should be 0 Hz.

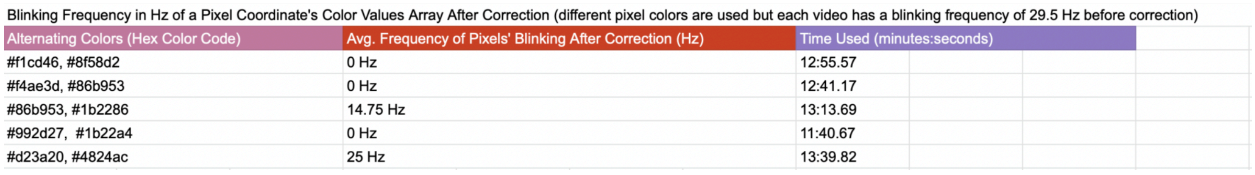
Challenges & Technical Issues

As is the case for the development of any other problem-solving mechanisms, the creation of the proposed software workflow did not come without challenges. From the beginning, it was decided that machine learning, particularly tabular (time series) classification, would be implemented to recognize the blinking of pixels. The desired software would have to deconstruct video frames into individual pixel coordinates, where each pixel coordinate's sequence of colors throughout the frames would be recorded in an array/set. The machine learning model, trained on such features, would possess the ability to recognize whether a particular set contained blinking activity or not. This proposal initially seemed feasible until a new scenario arose: what would happen if a set represented blinking in its first half, but its second half represented a completely idle, white pixel? The model may predict that the entire set is blinking, causing the program to correct the entire video even though safe, idle displays existed. To preserve non-blinking values, only the blinking segments needed to be detected and corrected, thus raising an unforeseen need for problem solving. The resultant solution incorporated a loop which iterated through the set containing the pixel's color values. At each iteration, the check index was incremented, and the trained machine learning model would determine whether or not all values until the check index represented blinking. If the model determined that these values were blinking, it would correct them and prevent them from being corrected again during the operation's lifetime. If the model determined that the presented values were not blinking for more than eighty iterations (representing approximately 1.33 seconds, considering that each video should display sixty frames per second), these eighty values would be preserved as well. Altogether, this new technique corrected intervals of the set representing blinking, and preserved intervals which did not represent blinking.

Another major challenge which arose during the workflow's development included its time-consuming manner. To correct a four-by-four pixel, two second video the program may run for nearly twenty-five minutes on the free, standard Google Colaboratory platform. Thus, data collection had to remain small-scale to finish in reasonable amounts of time, and greater computing power to crunch more demanding videos.

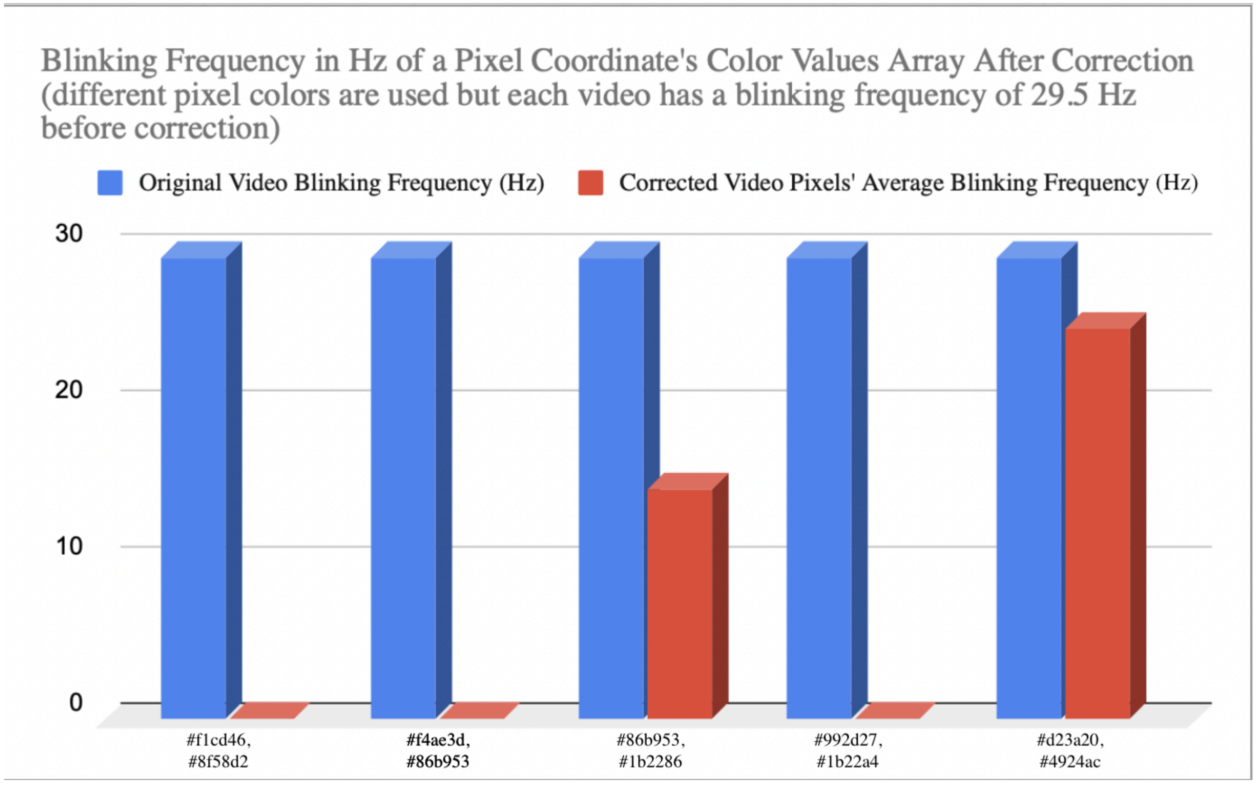
To date, the workflow's creation required the overcoming of several challenges, although room for improvement remains ever-present, especially concerning the lengthiness of its running time.

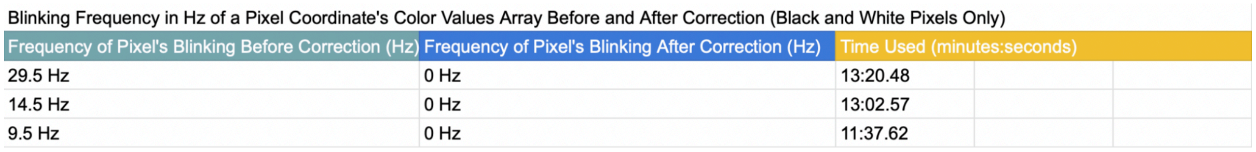
Experiment Results



Blinking Frequency in Hz of a Pixel Coordinate's Color Values Array After Correction (different pixel colors are used but each video has a blinking frequency of 29.5 Hz before correction)

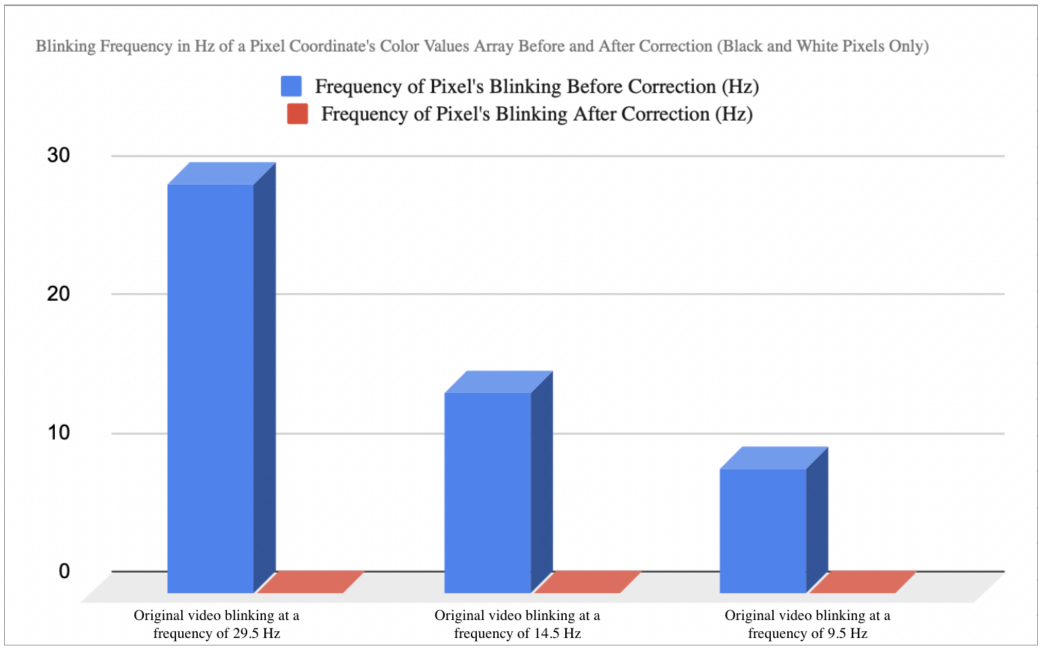
Alternating Colors (Hex Color Code)	Avg. Frequency of Pixels' Blinking After Correction (Hz)	Time Used (minutes:seconds)
#f1cd46, #8f58d2	0 Hz	12:55.57
#f4ae3d, #86b953	0 Hz	12:41.17
#86b953, #1b2286	14.75 Hz	13:13.69
#992d27, #1b22a4	0 Hz	11:40.67
#d23a20, #4824ac	25 Hz	13:39.82



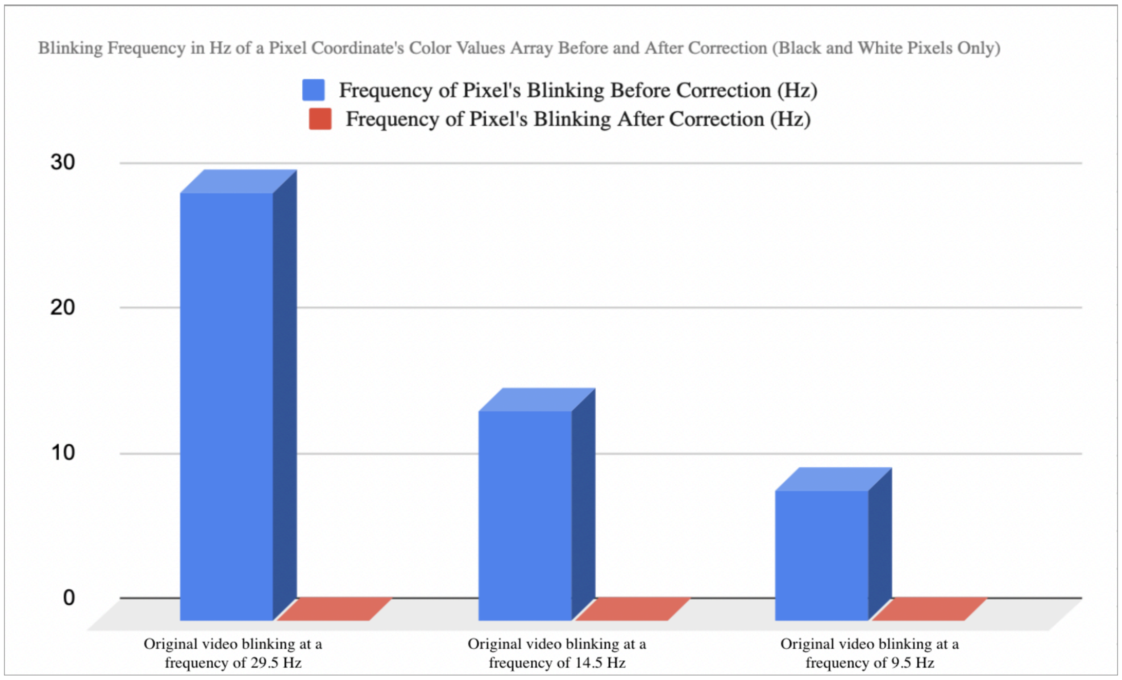


Blinking Frequency in Hz of a Pixel Coordinate's Color Values Array Before and After Correction (Black and White Pixels Only)

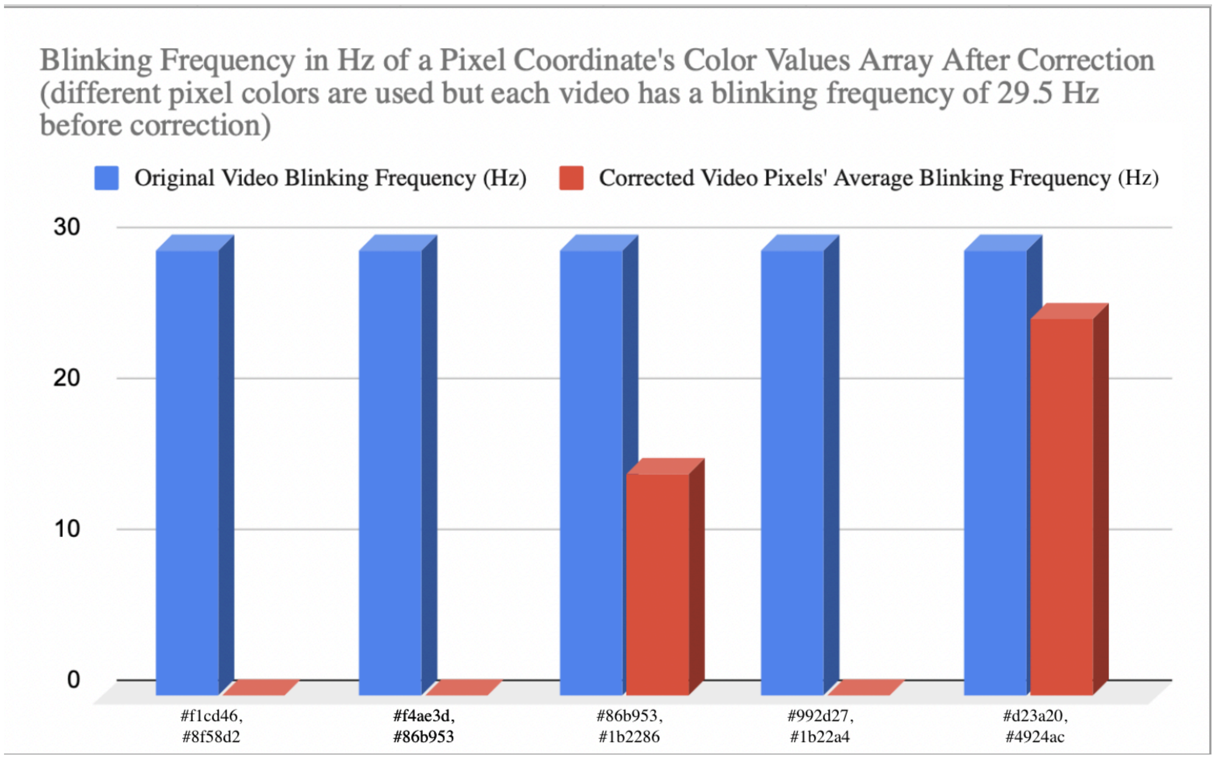
Frequency of Pixel's Blinking Before Correction (Hz)	Frequency of Pixel's Blinking After Correction (Hz)	Time Used (minutes:seconds)
29.5 Hz	0 Hz	13:20.48
14.5 Hz	0 Hz	13:02.57
9.5 Hz	0 Hz	11:37.62



Data Analysis and Discussion



Upon the completion of the proposed workflow's development, the workflow's capabilities were evaluated using a simple test: given a video that runs for a single second, would the final product be competent at correcting pixels that blink at various frequencies? Variables such as video length and involved pixel colors (black and white only) would remain consistent to obtain the best possible data where only a single variable was manipulated each time. The manipulated variable variable was the frequency in hertz (Hz) which blinking took place at in each video. The variable was manipulated thrice, with one video blinking between black and white pixels at 29.5 Hz, the next blinking at 14.5 Hz, and the final simple test blinking at 9.5 Hz. After allowing the workflow to recognize blinking and correct it in the three monochrome videos, data collected displayed that each output video blinked at 0 Hz. Because hertz as a unit measures the number of cycles (the number of times the color changes, and returns to the original color in the case of this workflow) which occur per second, blinking in the output videos was essentially eliminated completely. Because the correction of pixel color values is determined by the workflow’s ability to detect instances of blinking using machine learning, this proves the trained binary classification model to be successful. It competently understands what a sequence of black-white blinking pixel values looks like. However, because these initial tests were limited to monochrome pixels, the consistent success may have been caused by a sufficient number of black-white training data values. However, the workflow took an average of twelve minutes and thirty-seven seconds to correct an entire video. The lengthiness of the time required for the workflow to correct a video may be derived from needing to decompose the input video into pixel coordinate locations and iterate through each location’s values throughout frames to detect blinking. Several iteration loops are implemented, a probable cause of the workflow’s lifetime spanning over ten minutes for four-by-four pixel, sixty frame video data. The calculated graph created for these three tests is a column chart, or bar graph, which compares the blinking frequency of the original video against the blinking frequency of the corrected video per trial. Two bars are present per trial, one displaying the blinking frequency before correction, and one after. The x-axis displays the original blinking frequency which corresponds to each pair of bars.



To diversify tests, thus obtaining a more wholesome understanding of the model's capabilities, testing was then done on non-monochromatic video data. Control variables in these tests included the frequency at which videos blinked (29.5 Hz) and the length of each input video. The manipulated variable was the colors which the videos blinked between, while the dependent variable was the average frequency at which the output video's pixels blinked. After collecting data, blinking between #f1cd46 and #8f58d2 (yellow and purple), #f4ae3d and #86b953 (orange and green), as well as #992d97 and #1b22a4 (magenta and blue) was completely eliminated in the output videos: the output videos blinked at 0 Hz (no blinking). However, two other test cases, videos containing blinking between #86b953 and #1b22a4 (orange and blue) as well as #d23a20 and #4924ac (orange-red and a darker blue/purple) did not fare as well after being corrected by the model. The orange and blue blinking video began at the default blinking frequency of 29.5 Hz, but its output yielded an average blinking frequency of 14.75 Hz. The blinking between #d23a20 and #4924ac did not do as well either, with its output video having a blinking frequency of 25 Hz. In both of the less successful tests, some blinking was corrected, but not all, seeing to it that neither of them blink at 0 Hz. Possible causes for this lack of success may be the trained machine learning model's familiarity with these colors and shades. Each of these tests' original videos blinked at 29.5 Hz, and some tests were extremely successful in comparison to the two in question. Thus, frequency may be ruled out as a factor which drastically throws off the machine learning model, the workflow's detection guts. Color, the manipulated variable, must play a role in the success of detection. In other words, some tests may have been successful, because training data familiarized the model with such color values, while some were less consistent in correction, as the model was not sufficiently trained using such values. This displays that while the workflow is successful at correcting most of the videos and colors in the second set of tests, its machine learning particle may benefit from more diverse training data. The calculated graph created for these three tests is a column chart, or bar graph, which compares the blinking frequency of the original video against the blinking frequency of the corrected video per trial. Two bars are present per trial, one displaying the blinking frequency before correction, and one after. The x-axis displays colors involved in each original video.

Upon the proposal of the developed workflow, the following question was asked, "Is it possible to automate the creation of harmful videos' safer variations, particularly using machine learning methods?" Altogether, the collected data answers this question with a "yes," displaying that the entire workflow, which consists of both machine learning and some algorithmic thinking, was successful at correcting blinking videos. Because blinking videos are hazardous to individuals with conditions such as photosensitive epilepsy, this workflow makes a step towards creating a 'movie/video seatbelt' for people with special needs. The data collected proves that the workflow indeed corrects blinking videos.

Conclusion

Answering the initial testable question, “Is it possible to automate the creation of harmful videos’ safer variations, particularly using machine learning methods?” collected data displays an absolute “Yes!” The testing and evaluating of the developed workflow displays that machine learning coupled with basic algorithmic thinking indeed has the ability to make video data more accessible for certain members of the population. A couple types of trials were run in an attempt to evaluate the workflow’s capabilities, each yielding satisfactory results. The first type of testing included the manipulation of original videos’ blinking frequency as the independent variable, with the dependent variable being the output blinking frequency. For the three trials run in this first set, the output videos after 29.5 Hz, 14.5 Hz, and 9.5 Hz data were inputted had no blinking whatsoever (0 Hz). These three test inputs were all monochrome, four-by-four pixel, minute-long videos. Because in each test, blinking, the object of this project, was completely corrected, these trials reflected success. The second set of trials implemented four-by-four pixel, minute-long videos blinking at 29.5 Hz, but colors involved were now the independent variable. Most test cases yielded complete correction of blinking (0 Hz), although some yielded less consistent correction, possibly due to the model’s unfamiliarity with such colors. Altogether, the workflow made a step forward in correcting harmful video stimuli, blinking, using machine learning, making a step forward in increasing accessibility for individuals with photosensitive epilepsy.

Recommendations

For any individuals attempting research in machine learning, particularly machine learning related to the behavior or pixels represented tabularly, the best receivable suggestion may be to diversify training data. When training data is limited to certain colors or intervals, the trained model may have an acceptable ability to recognize particular patterns, but it may be less viable when such patterns are stretched out across color values training has not yet touched on. When the model developed in this project was tasked with detecting white-black blinking, which includes colors familiar from training, correction was more consistent and accurate. However, when tasked with detecting blinking between two unfamiliar color shades, the model could occasionally make corrections (it has some understanding of patterns), but less consistently. Thus, increase the amount of data used, and ensure they are diverse to prevent the model from becoming skewed. Choosing more optimal models may also allow for better understanding of the now increased data.

Furthermore, from experience gained in this project's development, it is suggested that the efficiency of one's workflow is optimized. To ultimately build atop the product of this project, the entire program which corrects hazardous blinking activity in videos must be made more quick and streamlined for practicality. Currently, because a mere two second, four-by-four pixel video may require up to twenty minutes to finish operations on, the correction of higher resolution and longer videos may grow unreasonably time-consuming. Thus, to make the accessibility software proposed in this project more accessible to everyone in a speedy manner, processes must be improved. In particular, the current bottleneck of the program is the method which thoroughly detects which intervals of an array containing a pixel's color values are blinking. Because the program should correct intervals where blinking is detected only, not touching idle parts of the video, a method implementing loops is employed. However, this method is time consuming, thus making research into detecting only blinking intervals in a more timely manner suggested. Moreover, the current method independently observes the values of individual pixels to recognize blinking. The use of methods such as image segmentation may allow the workflow to understand an image's different parts, thus optimizing correction color choice. The correction of other types of stimuli harmful to photosensitive individuals may be worthy of research as well. These subjects include particularly dangerous colors or luminescence and the correction of less consistent blinking, perhaps in relation to neighboring pixels' patterns as well.

The current product developed in this project already has the capability of correcting blinking in videos harmful to photosensitive peoples. This progress is the first step in the direction towards increasing internet accessibility and video-induced injury, and with enough additional research, these developments may one day become more mature accessibility tools.

Acknowledgements

To make the research described in this project possible, much appreciation goes to school faculty members who organize the annual science fair which encourages research, experimentation, and curiosity within our student body. The fair provides motivation and support for students pursuing their own discoveries, and opens up the opportunity for more experienced individuals to hear about one's findings. Furthermore, many thanks also go to the researchers and pioneers who have advanced the cutting-edge of science for centuries. Without the mathematical and statistical theories, machine learning algorithms, and end-to-end frameworks prior scientists have developed, this project's research, which builds atop existing technologies, would be impossible. The open-source code which developers share online, publications released by industry experts, and documentation published by software companies have made obtaining reliable, trustworthy information possible over the internet. Existing technologies have made the programming of the developed workflow feasible, and publications have made obtaining needed instructions possible at the click of a button.

Many thanks must also go to my parents, who have for many years facilitated the growth of my problem solving and basic technical skills. The education they have provided since my earliest years have equipped me with the skills that made working on this project possible. May all praise be to the Creator who my breath, health, and ability in researching would not have been possible without.

Work Cited

Tychsen L, Thio LL. “Concern of Photosensitive Seizures Evoked by 3D Video Displays or Virtual Reality Headsets in Children: Current Perspective”. Eye Brain. PubMed Central, National Library of Medicine. 2020 Feb 11. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7023866/>.

Marks, Hedy. “Photosensitive Epilepsy”. WebMD. 20 Nov. 2022. <https://www.webmd.com/epilepsy/guide/photosensitive-epilepsy-symptoms-causes-treatment>.

“Python OpenCV: Splitting Video into frames”. techtutorialsx. n.d. <https://techtutorialsx.com/2021/04/29/python-opencv-splitting-video-frames/>.

“tfdf.keras.GradientBoostedTreesModel”. TensorFlow. n.d. <https://www.tensorflow.org/decision_forests/api_docs/python/tfdf/keras/GradientBoostedTreesModel>.

hfawaz. “Timeseries classification from scratch”. Keras. 16 Jul. 2020.
<https://keras.io/examples/timeseries/timeseries_classification_from_scratch/>.